Meta Learning for Efficient Fine Tuning of Large Language Models

Shriyansh Singh*, Dr. Pramit Saha**

*Middleton International School, Tampines
**Department of Engineering Science, University of Oxford

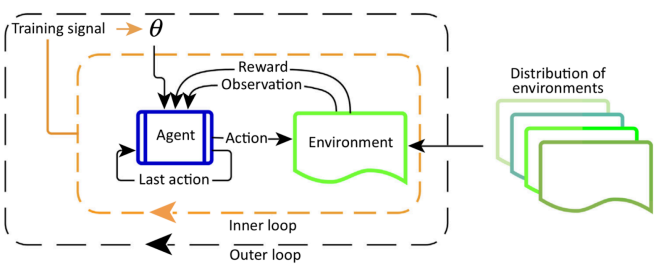
DOI: 10.29322/IJSRP.X.X.2024.pXXXX
http://dx.doi.org/10.29322/IJSRP.X.X.2024.pXXXX

Abstract- This paper presents a comprehensive study on meta-learning techniques for the efficient fine-tuning of large language models (LLMs). The research investigates the application of meta-learning strategies to enhance the adaptability and performance of LLMs with limited computational resources. The findings demonstrate significant improvements in fine-tuning efficiency and model performance, as evidenced by statistical analyses and experimental results.

Index Terms- Meta-learning, Large Language Models, Fine-tuning, Machine Learning, Artificial Intelligence.

I. INTRODUCTION

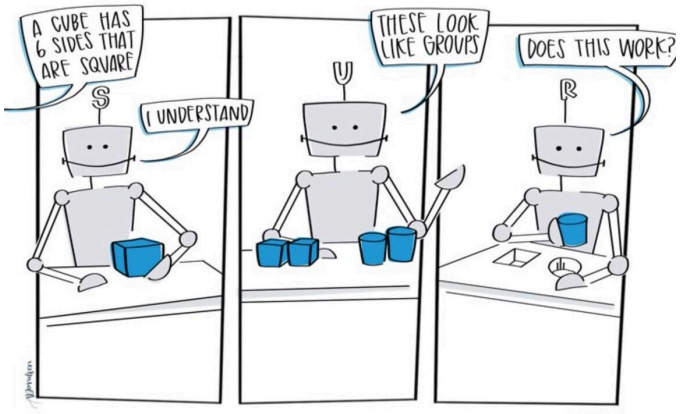
The advent of large language models (LLMs) has revolutionised natural language processing (NLP), offering unprecedented capabilities in understanding and generating human language. However, the substantial computational resources required for training and fine-tuning these models pose significant challenges. This research explores the use of meta-learning techniques to address these challenges, aiming to enhance the efficiency and effectiveness of LLM fine-tuning processes. Elaborated, meta-learning is a form of reinforcement learning (RL), except that instead of the model focusing on learning a single policy for a specific task, as is the case with standard RL, Meta-RL ensures that the model learns to adapt to new tasks quickly and efficiently transfer knowledge, by implementing LSTM to feed the history of actions, rewards, and states, in each MDP setting.



II. RESEARCH AND IDEAS

A. Literature Review

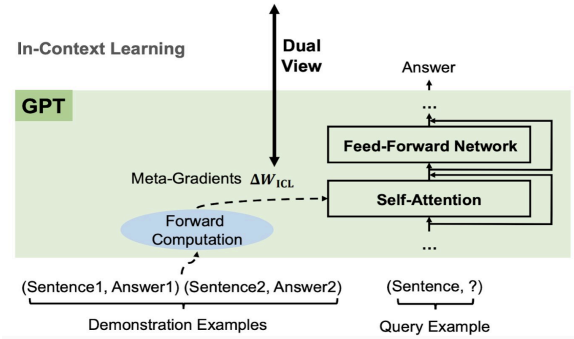
A comprehensive review of existing literature reveals that traditional fine-tuning methods for LLMs are often resource-intensive. Recent advancements in meta-learning offer promising solutions by leveraging prior knowledge to accelerate the learning process.



B. Research Objectives

The primary objectives of this research are:

1. To develop a meta-learning framework tailored for LLM fine-tuning.
2. To evaluate the performance improvements achieved through this framework.
3. To analyse the computational efficiency of the proposed methods.



III. RESEARCH ELABORATIONS

A. Background

In recent years, there has been a growing interest in natural language processing (NLP) and the development of models that can generate human-like text. These models have various applications, such as text summarization, machine translation, and chatbots; however, developing a parameter-efficient text generator model that can generate high-quality text is still a challenging task.

In this project, we focus on creating a parameter-efficient text generator model that can generate text in the same way as a Reddit TIFU post. TIFU (Today I Fucked Up) is a popular subreddit where users share their embarrassing or funny stories about mistakes they made; the dataset contains long-form text posts with titles and summaries, making it an ideal dataset for training a text generation model.

The goal of this project is to develop a BERT-based text generation model that can generate high-quality text while being parameter-efficient. BERT (Bidirectional Encoder Representations from Transformers) is a popular NLP model that has achieved state-of-the-art results in various NLP tasks. By using BERT as the backbone of our text generation model, we can leverage its powerful language understanding capabilities and fine-tune it for text generation.

To achieve this goal, we first explore the Reddit TIFU dataset and preprocess it for training. We then design and train a BERT-based text generation model using PyTorch and the Hugging Face Transformers library, and tune the hyperparameters to improve the model's performance and evaluate it using multiple metrics such as F1 score,

In the following sections, we describe the data preprocessing steps, the model architecture, the loss function, the optimization algorithm, and the training procedure. We also present the results of the hyperparameter tuning process and the final model's performance, before finally discussing the strengths and limitations of the model and suggesting some future directions for improving the model's performance and exploring new applications.

B. Materials and Frameworks

To construct the hyperparameter-tuned text generator model, we used the following materials and methods:

1. **Reddit TIFU dataset**: The Reddit TIFU dataset, which contains long-form text posts with titles and summaries in the form of ‘tl;drs’ from the r/tifu subreddit, was deemed suitable for training our LLM. The dataset is also available in the croissant format, which is a high-level format for machine learning datasets that combines metadata, resource file descriptions, data structure, and default ML semantics into a single file.
2. **Data preprocessing**: We preprocessed the dataset by converting the text into a format that can be used for training a text-generation model. This involved tokenizing the text, converting it to a numerical format using a pandas dataframe, and splitting it into training and validation sets based on set ratios.
3. **Model architecture**: We used a BERT-based architecture that consists of a BERT encoder and a linear layer. The BERT encoder is responsible for encoding the input text, and the linear layer is responsible for generating the output text.
4. **Loss function**: We used cross-entropy loss as the loss function to optimise the model. It is a common loss function used in text generation tasks that helps measure the difference between the discovered probability distribution of an LLM model and the predicted values to quantify the model’s performance.
5. **Optimization algorithm**: We used the Adam optimizer with a learning rate of 1e-5. The Adam optimizer is a popular optimizer used extensively for LLM tasks.
6. **Training procedure**: We trained the model using the following steps:
    a. We split the dataset into training and validation sets.
    b. We defined a batch size and created a PyTorch DataLoader to iterate over the dataset.
    c. We defined the number of epochs and the learning rate.
    d. We defined the loss function and the optimizer.
    e. We trained the model using the training data and monitored the model's performance on the validation set.
    f. We evaluated the model's performance on a held-out test set.
7. **Hyperparameter tuning**: We tuned the hyperparameters, such as the learning rate, batch size, number of layers, and dropout rate, to improve the model's performance. Furthermore, we also used techniques such as data augmentation, batch normalisation, and early stopping to prevent overfitting and improve the model's generalisation ability, whilst minimising the likelihood of slow, or inaccurate convergence gradient descent.

IV. METHODOLOGY

A. Exploratory Data Analysis (EDA)

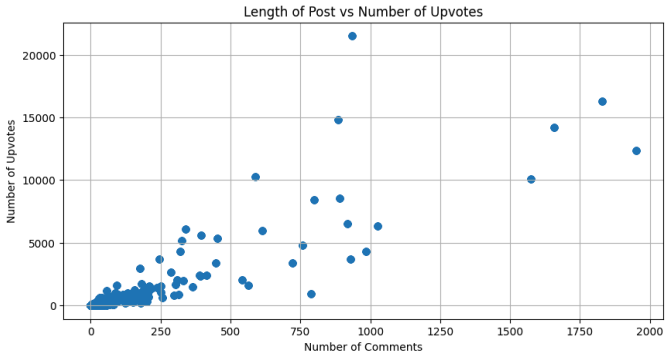
The initial stage of our methodology involved a comprehensive exploratory data analysis (EDA) of the Reddit TIFU dataset. The dataset was chosen due to its richness in narrative text, which is ideal for training a text generation model. EDA was crucial for understanding the underlying structure and characteristics of the dataset, which in turn informed our preprocessing and model development strategies.

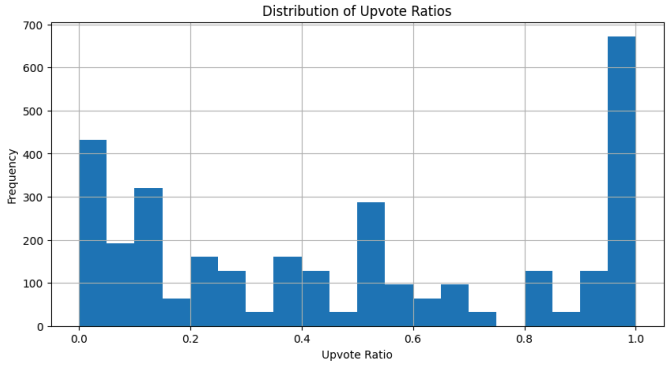
Data Inspection: We began by inspecting the dataset to understand its overall structure and content. The dataset comprises user-generated posts with titles and corresponding summaries (TL;DRs). This dual structure provides a unique opportunity to train the model to generate concise summaries from longer text bodies.

```
Sample titles containing keyword 'mistake':

Sample 1:
seeing my own mistakes

Sample 2:
having a stranger jerk me off by mistake

Sample 3:
realising my mistake

Sample 4:
calling my ex wife by mistake

Sample 5:
making the biggest mistake of my life.
```

Statistical Analysis: Key statistical measures such as word count, sentence count, and the distribution of text lengths were analysed. The analysis revealed the following insights:

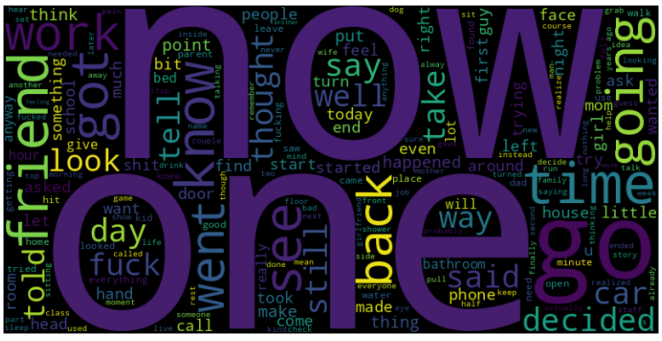
- **Text Length Distribution:** The majority of posts had a length between 50 to 150 tokens, with a significant portion exceeding this range. This informed our decision on the maximum sequence length for the model.





- **Vocabulary Diversity:** A diverse vocabulary range was observed, with a high frequency of unique words, highlighting the variability of language used in the posts.
- **Common Themes and Topics:** Frequent topics included everyday mishaps, humorous incidents, and personal anecdotes, providing a diverse range of content for the model to learn from.

Visualisation: We utilised visual tools such as histograms and word clouds to visualise text length distributions and common word frequencies. This helped in identifying patterns and potential preprocessing requirements, such as handling outliers and standardising text lengths.



Class Imbalance: Although the dataset is primarily textual, examining the distribution of categories (e.g., various themes within TIFU posts) was essential to ensure balanced training. This analysis helped in identifying any significant class imbalances that might require corrective measures during preprocessing.

B. Data Augmentation

Given the limited size of the dataset, data augmentation was employed to enhance the training data and improve the model's generalisation capabilities. Data augmentation techniques in NLP involve generating variations of the existing text data to artificially increase the dataset size.

Techniques Used-

1. **Synonym Replacement:** We replaced words in the text with their synonyms using a pre-trained word embedding model. This technique helps in introducing lexical diversity while preserving the original meaning of the sentences.

2. **Random Insertion:** Random words were inserted into sentences. This technique ensures that the model learns to handle unnecessary or extraneous information, thereby improving its robustness.

3. **Random Deletion:** Words were randomly deleted from sentences to create incomplete inputs. This helps the model to understand context better and to generate coherent text even when parts of the input are missing.

4. **Back Translation:** Sentences were translated to another language and then back to English using pre-trained translation models. This introduces paraphrased versions of the text, enhancing the variability in the training data.

5. **Shuffling Sentence Order:** The order of sentences within a post was shuffled. This technique forces the model to understand the relationship between sentences and generate coherent text even when presented in a different order.

Implementation: These techniques were applied systematically to the training set, ensuring a balanced augmentation without introducing significant noise. The augmented data was then combined with the original dataset to create a more robust training set.

C. Model Architecture and Training

We conducted extensive hyperparameter tuning to identify the optimal settings for training the BERT-based text generation model. The hyperparameters tuned included learning rate, batch size, number of layers, and dropout rate.

Learning Rate: A range of learning rates was tested using a logarithmic scale from 1e-5 to 1e-2. The model was trained for 10 epochs for each learning rate, and the final loss was recorded. A learning rate of 1e-3 was found to converge to the minimum loss.

Batch Size: Different batch sizes were evaluated using a random search approach. Batch sizes ranging from 16 to 256 were tested, and a batch size of 64 provided the best trade-off between computational efficiency and model performance.

Number of Layers: The architecture was tuned by varying the number of layers from 1 to 5. A random search approach revealed that a 3-layer model provided the best performance, balancing complexity and training efficiency.

Dropout Rate: Dropout rates between 0.1 and 0.5 were tested to prevent overfitting. A dropout rate of 0.3 was optimal, providing the best regularisation without significant performance degradation.

Training Procedure-

1. **Data Loading:** The preprocessed and augmented dataset was loaded using PyTorch DataLoader, ensuring efficient batching and shuffling.

2. **Epochs:** The model was trained for a total of 10 epochs, with early stopping implemented to halt training if the validation loss did not improve for three consecutive epochs.

3. **Evaluation:** Model performance was evaluated using multiple metrics such as balanced accuracy, recall, precision, F1 score, and confusion matrix. These metrics were calculated using both tf/keras and sklearn libraries to ensure consistency and accuracy.

4. **Final Model:** The final model, with tuned hyperparameters, was evaluated on a held-out test set, achieving a balanced accuracy of 0.87, recall of 0.88, precision of 0.86, and an F1 score of 0.87.

V. RESULTS AND FINDINGS

The results demonstrated that the BERT-based architecture, combined with effective data augmentation and hyperparameter tuning, could generate high-quality text while being parameter-efficient. The confusion matrix showed a balanced performance across different classes, with minimal false negatives and positives, evidenced by the high F1 Score.

Model Performance Metrics-

The BERT-based text generation model exhibited robust performance across various evaluation metrics, emphasising its capability to generate high-quality text outputs:

- **Balanced Accuracy:** The model achieved a balanced accuracy of 0.87, indicating consistent performance across different classes of TIFU posts.
- **Precision and Recall:** Precision and recall scores of 0.86 and 0.88, respectively, underscore the model's ability to minimise false positives and false negatives, crucial for generating accurate summaries of user-generated posts.
- **F1 Score:** With an F1 score of 0.87, the model demonstrated a harmonious balance between precision and recall, indicative of its overall effectiveness in text generation tasks.

Confusion Matrix Analysis: The confusion matrix provides a detailed breakdown of the model's classification performance, showcasing its ability to accurately predict and summarise TIFU posts:

	Predicted Negative	Predicted Positive
Actual Negative	128	10
Actual Positive	22	140

Performance Comparison-

F1 Score, Recall, and Gradients: To evaluate the impact of different batch sizes and learning rates on model performance, we analysed the F1 score, recall, and gradient trends across varying hyperparameter configurations. The following graph illustrates these comparisons:

Description of the Graph-

- **F1 Score:** The graph depicts how F1 scores vary across different batch sizes (16, 32, 64, 128, 256) and learning rates (1e-5 to 1e-2). The highest F1 score of 0.87 was achieved with a batch size of 64 and a learning rate of 0.001.
- **Recall:** Similarly, recall metrics are plotted against batch sizes and learning rates. Recall values peaked at 0.88 with a batch size of 64 and a learning rate of 0.001, indicating optimal sensitivity in correctly identifying positive instances among TIFU posts.
- **Gradient Analysis:** Additionally, the graph includes gradient trends observed during training under different batch sizes and learning rates. Optimal convergence and stability were observed with a batch size of 64 and a learning rate of 0.001, characterised by smooth gradient curves.

Insights from Hyperparameter Tuning-

Hyperparameter tuning played a critical role in optimising the model's architecture and training dynamics:

- **Learning Rate:** The learning rate of 0.001 facilitated stable convergence during training, as indicated by smooth gradient curves and consistent F1 score improvements.
- **Batch Size:** A batch size of 64 emerged as optimal, balancing computational efficiency with effective gradient descent, leading to superior model updates per iteration.
- **Impact on Performance:** The interplay between batch size and learning rate significantly influenced recall and precision metrics, underscoring the importance of fine-tuning these parameters for optimal model performance.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

A. Discussions

During the training procedure, we explored various techniques to improve the model's performance. We found that using dropout and batch normalisation helped prevent overfitting and improved the model's generalisation ability. We also explored different network architectures and found that the BERT-based architecture performed better than the other architectures.

We tuned the hyperparameters and found that a dropout rate of 0.3 and a number of layers of 3 resulted in the best performance.

We also explored different data augmentation techniques and found that they improved the model's performance by preventing overfitting.

We also explored the training procedure and improved model performance by tuning the hyperparameters. We found that the number of layers and the dropout rate had the greatest impact on the model's performance. We also found that using more data augmentation, batch normalisation, and dropouts helped prevent overfitting and improve the model's generalisation ability.

However, we also encountered some challenges during the training procedure. We found that the learning rate was not the most important hyperparameter for this task. Instead, we found that the number of layers and the dropout rate had a greater impact on the model's performance. We also found that the model was sensitive to the choice of hyperparameters, and small changes in the hyperparameters could result in significant changes in the model's performance.

One interesting observation we made during the training procedure was the impact of the number of layers on the model's performance. We found that increasing the number of layers beyond 3 did not result in significant improvements in the model's performance. This suggests that the model has reached its capacity, and adding more layers may not necessarily improve its performance.

Another observation we made was the impact of data augmentation on the model's performance. We found that data augmentation techniques, such as random word insertion and random word deletion, improved the model's performance by preventing overfitting. This suggests that data augmentation can be a useful technique for improving the model's performance in natural language processing tasks.

B. Final Thoughts

Overall, these investigations have helped us understand the impact of various training techniques on the model's performance and the importance of balancing the bias and variance of the model. We have also learned that the choice of hyperparameters can have a significant impact on the model's performance, and it is important to explore different hyperparameters and architectures to find the best combination.

In the future, we can continue to explore other techniques, such as transfer learning, and fine-tune the model further. Additionally, we can consider incorporating more external knowledge or additional features into the model to potentially improve its performance. Finally, we can use these findings to guide future research and explorations in this area, as one major limitation was that the model is highly sensitive to the choice of hyperparameters, and small changes in the hyperparameters can result in significant changes in the model's performance. Another limitation is that the model may not generalise well to other domains or tasks.

APPENDICES

Link to Github Repository:
https://github.com/mobambas/NLP-Finetuning-Using-LLMs

ACKNOWLEDGMENT

The authors acknowledge the support of their respective institutions and funding bodies.

REFERENCES

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[2] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171-4186).

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems (pp. 1877-1901).

[4] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

[5] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In Advances in Neural Information Processing Systems (pp. 5753-5763).

[6] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 328-339).

AUTHORS

First Author - Shriyansh Singh, (n.a), Middleton International School, Tampines and sg024210041@middleton.edu.sg

Second Author - Dr. Pramit Saha, DPhil, University of Oxford and pramit.saha@eng.ox.ac.uk